PVRVFrame

User Manual

Copyright © Imagination Technologies Limited. All Rights Reserved.

This publication contains proprietary information which is subject to change without notice and is supplied 'as is' without warranty of any kind. Imagination Technologies and the Imagination Technologies logo are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.

Filename	:	PVRVFrame.User Manual
Version	:	PowerVR SDK REL_3.5@3523383a External Issue
Issue Date	:	17 Apr 2015
Author	:	Imagination Technologies Limited

Contents

1. Introduction 2

2. Getting Started 4

2.1. Hardware Requirements 4
2.2. Package Contents 4
2.2.1. Windows 4
2.2.2. Linux 5
2.2.3. OS X 5
2.3. Installation 5
2.3.1. Windows 5
2.3.2. Linux 5
2.3.3. OS X 6

3. PVRVFrame Graphical User Interface 7

3.1. Working with Profiles 7
3.2. Running an Application 9

4. Supported Extensions 10

5. Current Limitations 12

6. Contact Details 13

List of Figures

Figure 1. PVRVFrame layer overview 3
Figure 2. PVRVFrame GUI 7
Figure 3. Viewing further information 8
Figure 4. Preferences dialog box 8
Figure 5. Running an application 9

List of Tables

Table 1. Emulation requirements 4
Table 2. Package contents for Windows 4
Table 3. Package contents for Linux 5
Table 4. Package contents for OS X 5
Table 5. Supported EGL extensions 10
Table 6. Supported OpenGL ES extensions 10
Table 7. Current limitations 12

1. Introduction

PVRVFrame is an emulation layer for desktop platforms that provides an OpenGL ES and EGL environment mimicking that available on PowerVR-enabled devices. It works by redirecting OpenGL ES API calls to the underlying OpenGL desktop implementation present on the development system. PVRVFrame is aimed at developers writing OpenGL ES applications for PowerVR-enabled embedded devices and is designed to mitigate the need to work directly with this hardware. Figure 1 captures this understanding.

This utility is not pixel accurate, nor a performance simulator. The performance obtained when running OpenGL ES applications with PVRVFrame is not indicative of the performance obtained when running on real consumer hardware. It only gives a qualitative preview of an OpenGL ES application.

PVRVFrame is available for OpenGL ES 1.1, Common profile, OpenGL ES 2.0 and higher, and must be used with the corresponding version of the Khronos OpenGL ES header files. These header files should be downloaded from the Khronos website, although some versions are also bundled with PVRVFrame for convenience.

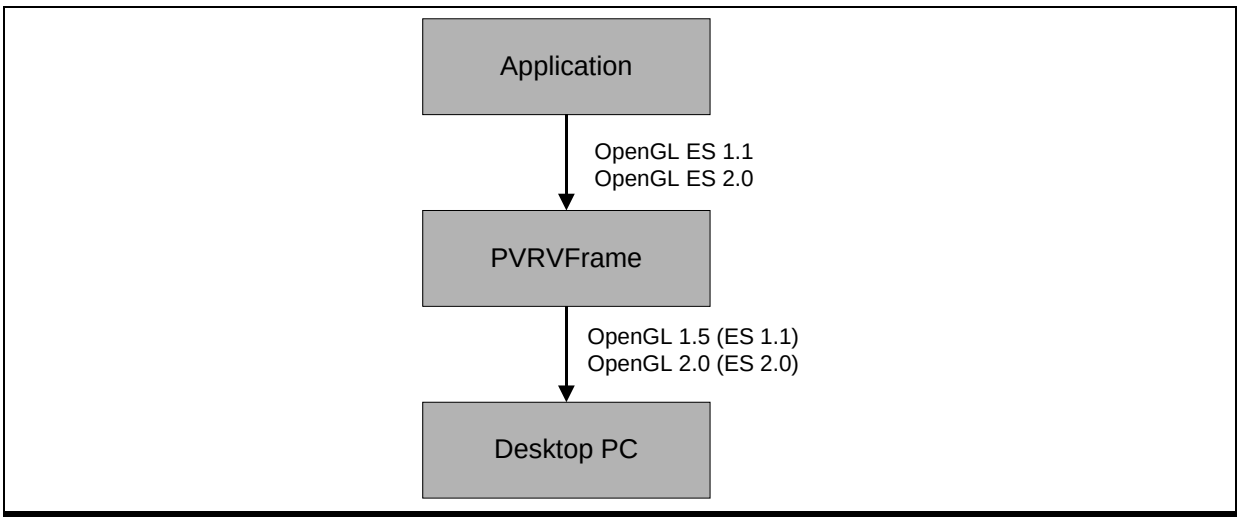

Figure 1. PVRVFrame layer overview

2. Getting Started

2.1. Hardware Requirements

PVRVFrame requires hardware compatible with the underlying OpenGL implementation of the version of OpenGL ES being targeted. Table 1 provides an indication of the requirements for emulation.

Table 1. Emulation requirements

Target	Requirements
OpenGL ES 1.1	OpenGL 1.5+
OpenGL ES 2.0	OpenGL 2.0+
OpenGL ES 3.0	OpenGL 3.3+
OpenGL ES 3.1	OpenGL 4.3+
`OES_packed_depth_stencil`	`EXT_packed_depth_stencil`
`OES_framebuffer_object`	`EXT_framebuffer object`
`OES_texture_float`	`ARB_texture_float`
`OES_texture_half_float`	`ARB_texture_float`
`OES_vertex_half_float`	`ARB_half_float_pixel`
`OES_vertex_array_object`	`ARB_vertex_array_object`
`EXT_shader_texture_lod`	`ARB_shader_texture_lod`
`IMG_multisampled_render_to_texture`	`EXT_framebuffer_multisample`

It should also be noted that in addition to these requirements, at least an Intel HD3000 with proper drivers is also necessary.

2.2. Package Contents

2.2.1. Windows

Table 2 lists the package contents for Windows.

Table 2. Package contents for Windows

Content	Description
`libEGL.dll`	PVRVFrame driver file for EGL 1.4.
`libGLES_CM.dll`	PVRVFrame driver file for OpenGL ES 1.1.
`libGLESv2.dll`	PVRVFrame driver file for OpenGL ES 2.0 and higher.
`libEGL.lib`	Import (“stub”) library for use when linking against `libEGL.dll`.
`libGLES_CM.lib`	Import (“stub”) library for use when linking against `libGLES_CM.dll`.
`libGLESv2.lib`	Import (“stub”) library for use when linking against `libGLESv2.dll`.

2.2.2. Linux

Table 3 lists the package contents for Linux.

Table 3. Package contents for Linux

Content	Description
libEGL.so	PVRVFrame driver file for EGL 1.4.
libGLES_cm.so	PVRVFrame driver file for OpenGL ES 1.1.
libGLESv2.so	PVRVFrame driver file for OpenGL ES 2.0 and higher.

2.2.3. OS X

Table 4 lists the package contents for OS X.

Table 4. Package contents for OS X

Content	Description
libEGL.dylib	PVRVFrame driver file for EGL 1.4.
libGLES_CM.dylib	PVRVFrame driver file for OpenGL ES 1.1.
libGLESv2.dylib	PVRVFrame driver file for OpenGL ES 2.0.

2.3. Installation

2.3.1. Windows

The DLL files that match the version of OpenGL ES that is to be targeted must be in a folder in the %PATH% environment variable or in the directory the application is running from. With the DLL files in place, a Visual Studio project can be set up as follows:

1. Create a new Visual C project, adding source and include files as necessary.
2. Add the following directory to the include paths of the project:

```
<SDKPackage>\Builds\Include\
```

3. Link against the supplied libraries. For OpenGL ES 1.1 projects, it is only required to link against libGLES_CM.lib. For OpenGL ES 2.0 and higher, it is required to link against both libEGL.lib and libGLESv2.lib.

2.3.2. Linux

To set up a new project to run with PVRVFrame, follow these steps:

1. Create a new makefile, adding source and include files as necessary.
2. Add the following directory to the include paths of the project:

```
<SDKPackage>\Builds\Include\
```

3. Link against the supplied libraries. For OpenGL ES 1.1 projects, it is only required to link against libGLES_CM.so. For OpenGL ES 2.0 and higher, it is required to link against both libEGL.so and libGLESv2.so.

4. Run the following terminal command:

```
export LD_LIBRARY_PATH=<lib folder>;$LD_LIBRARY_PATH
```

Note: It is possible to check that the correct versions of the libraries are being used and that the location of these libraries is set correctly. To do this, run the following command on an application built against the PVRVFrame libraries: `ldd <path to application>`

2.3.3. OS X

To set up a new project to run with PVRVFrame, follow these steps:

1. Create a new makefile, adding source and include files as necessary.
2. Add the following directory to the include paths of the project:

```
<SDKPackage>\Builds\Include\
```

3. Link against the supplied libraries. For OpenGL ES 1.1 projects, it is only required to link against `libGLES_CM.dylib`. For OpenGL ES 2.0 and higher, it is required to link against both `libEGL.dylib` and `libGLESv2.dylib`.

4. Run the following terminal command:

```
export DYLD_LIBRARY_PATH=<lib folder>;$DYLD_LIBRARY_PATH
```

Note: It is possible to check that the correct versions of the libraries are being used and that the location of these libraries is set correctly. To do this run the following commands on an application built against the PVRVFrame libraries: `otool -L <path to application>`

3. PVRVFrame Graphical User Interface

3.1. Working with Profiles

Profiles are pre-configured sets of extensions and capabilities. A selection of profiles is included with PVRVFrame. To select a given profile for PVRVFrame, perform the following steps:

1. Open the PVRVFrame GUI (Figure 2).

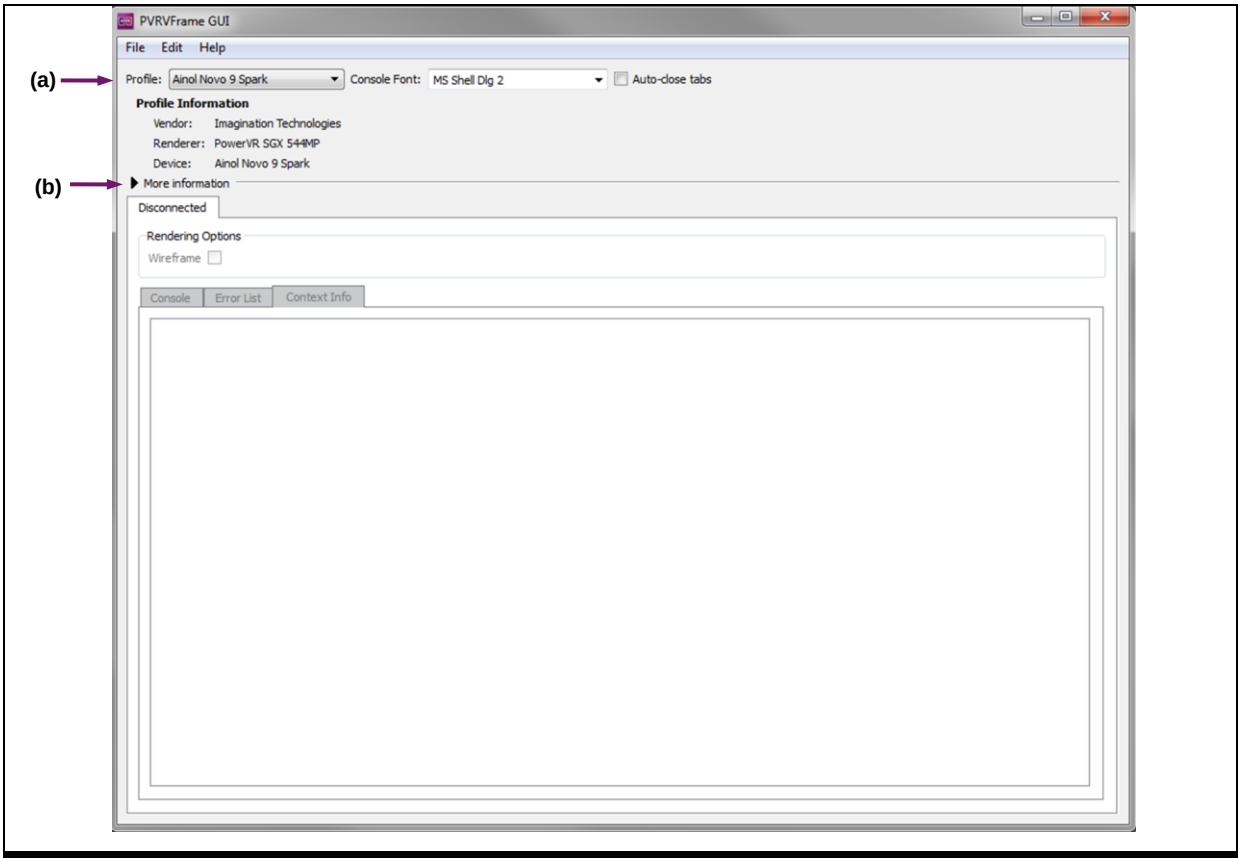

Figure 2. PVRVFrame GUI

2. From the `Profile` dropdown box, select the profile name that matches the device or chipset (Figure 2a).
3. Expand the option called `More Information` (Figure 2b) in order to view the capabilities, extensions and EGL configurations for the selected profile (Figure 3).

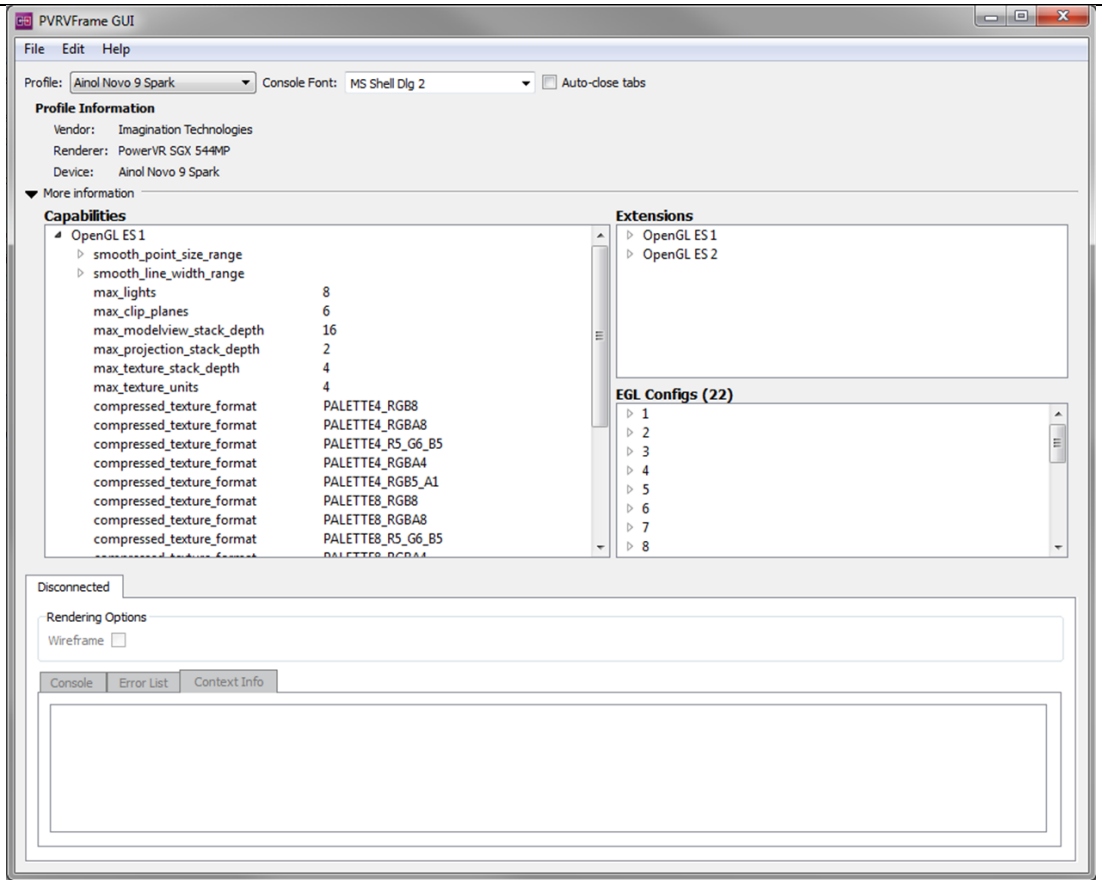

Figure 3. Viewing further information

Note: If no profiles are displayed in the `Profiles` *dropdown box, click* `Edit -> Preferences` *and ensure that the* `Directory` *field is set to the directory which contains the profile data files. The default directory for Windows is displayed in Figure 4.*

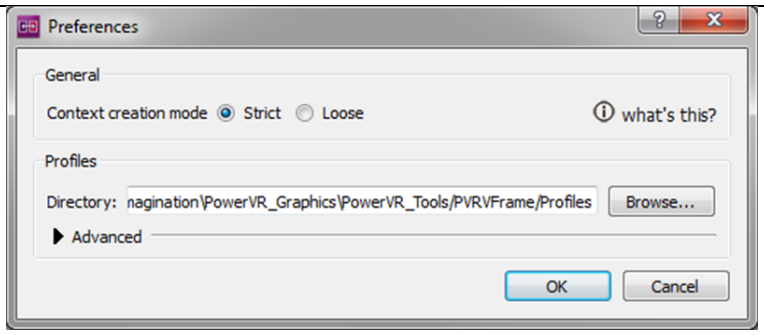

Figure 4. Preferences dialog box

3.2. Running an Application

When an OpenGL ES application is run (Figure 5a), it automatically connects to PVRVFrame and any GL errors or warnings that occur during the time that the application is run for will be displayed in the `Console` section of PVRVFrame (Figure 5b). By default, a running application will be displayed rendered. An application can also be rendered in wireframe by using the `Wireframe` checkbox under `Rendering Options` (Figure 5c).

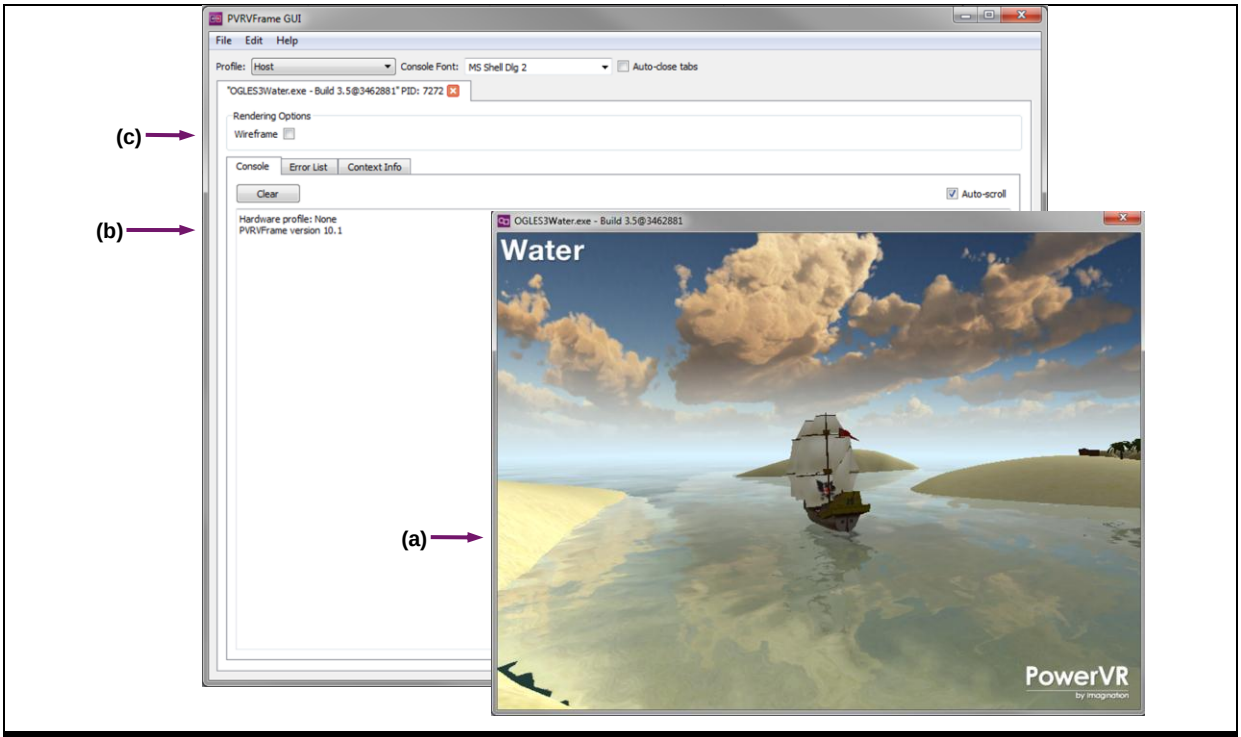

Figure 5. Running an application

Note: Other actions can also be achieved in PVRVFrame GUI such as changing the font used to display text in the console.

4. Supported Extensions

Table 5 and Table 6 provide lists of supported EGL and OpenGL ES extensions, respectively. Documentation for these extensions can be found in the Khronos registry at: http://www.khronos.org/registry/.

Table 5. Supported EGL extensions

Extension
EGL_KHR_fence_sync
EGL_IMG_context_priority
EGL_KHR_create_context

Table 6. Supported OpenGL ES extensions

Extension
APPLE_copy_texture_levels
APPLE_texture_2D_limited_npot
EXT_blend_minmax
EXT_debug_marker
EXT_discard_framebuffer
EXT_draw_buffers
EXT_multi_draw_arrays
EXT_multisampled_render_to_texture
EXT_occlusion_query_boolean
EXT_shader_texture_lod
EXT_texture_filter_anisotropic
EXT_texture_format_BGRA8888 (not supported on OS X)
EXT_texture_rg
EXT_texture_storage
EXT_texture_type_2_10_10_10_REV
IMG_multisampled_render_to_texture
IMG_program_binary
IMG_read_format
IMG_shader_binary
IMG_texture_compression_pvrtc
IMG_texture_compression_pvrtc2
IMG_texture_env_enhanced_fixed_function
IMG_texture_format_BGRA8888 (not supported on OS X)
IMG_texture_npot
IMG_texture_stream
IMG_texture_stream2

Extension
IMG_uniform_buffer_object
IMG_user_clip_plane
IMG_vertex_array_object
KHR_debug
OES_blend_equation_separate
OES_blend_func_separate
OES_blend_subtract
OES_byte_coordinates
OES_compressed_ETC1_RGB8_texture
OES_depth_texture
OES_depth_texture_cube_map
OES_draw_texture
OES_egl_sync
OES_element_index_uint
OES_extended_matrix_palette
OES_fixed_point
OES_fragment_precision_high
OES_framebuffer_object
OES_get_program_binary
OES_mapbuffer
OES_matrix_get
OES_matrix_palette
OES_packed_depth_stencil
OES_point_size_array
OES_point_sprite
OES_query_matrix
OES_read_format
OES_required_internalformat
OES_single_precision
OES_standard_derivatives
OES_stencil_wrap
OES_texture_cube_map
OES_texture_env_crossbar
OES_texture_float
OES_texture_half_float
OES_texture_mirrored_repeat
OES_vertex_array_object

5. Current Limitations

Table 1 provides a list of the current limitations of PVRVFrame.

Table 7. Current limitations

Aspect	Limitation
Precision qualifiers	PVRVFrame does not support precision qualifiers in shaders. Shaders using precision modifiers will still function but the precision qualifiers are ignored. All precisions are assumed to be `highp`.
Orthogonal Tex Env Combine	This is not supported.
PBuffer	This is not supported on OS X. It is supported on Windows and Linux.
Context	The same context cannot be used to render to two different windows under Windows. This functions correctly under Linux and OS X.
`glRenderBufferStorage`	Calling `glRenderBufferStorage` with `GL_STENCIL_INDEX8` does not work. `DEPTH24_STENCIL8`, which requires `OES_packed_depth_stencil`, should be used instead.

6. Contact Details

For further support, visit our forum:
http://forum.imgtec.com

Or file a ticket in our support system:
https://pvrsupport.imgtec.com

To learn more about our PowerVR Graphics SDK and Insider programme, please visit:
http://www.powervrinsider.com

For general enquiries, please visit our website:
http://imgtec.com/corporate/contactus.asp

Imagination Technologies, the Imagination Technologies logo, AMA, Codescape, Ensigma, IMGworks, I2P, PowerVR, PURE, PURE Digital, MeOS, Meta, MBX, MTX, PDP, SGX, UCC, USSE, VXD and VXE are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.